

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problems Mailbox.**

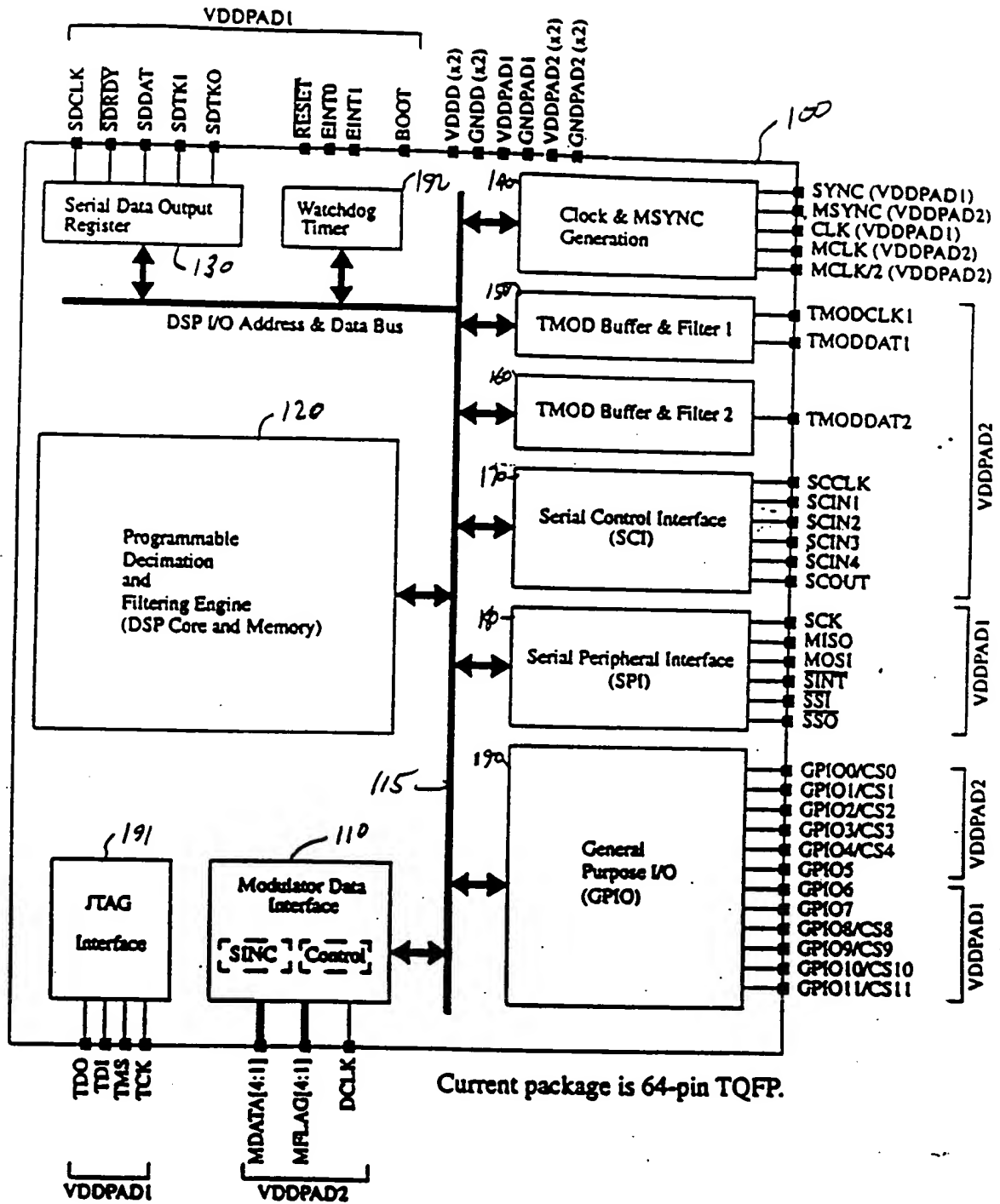


FIGURE 1

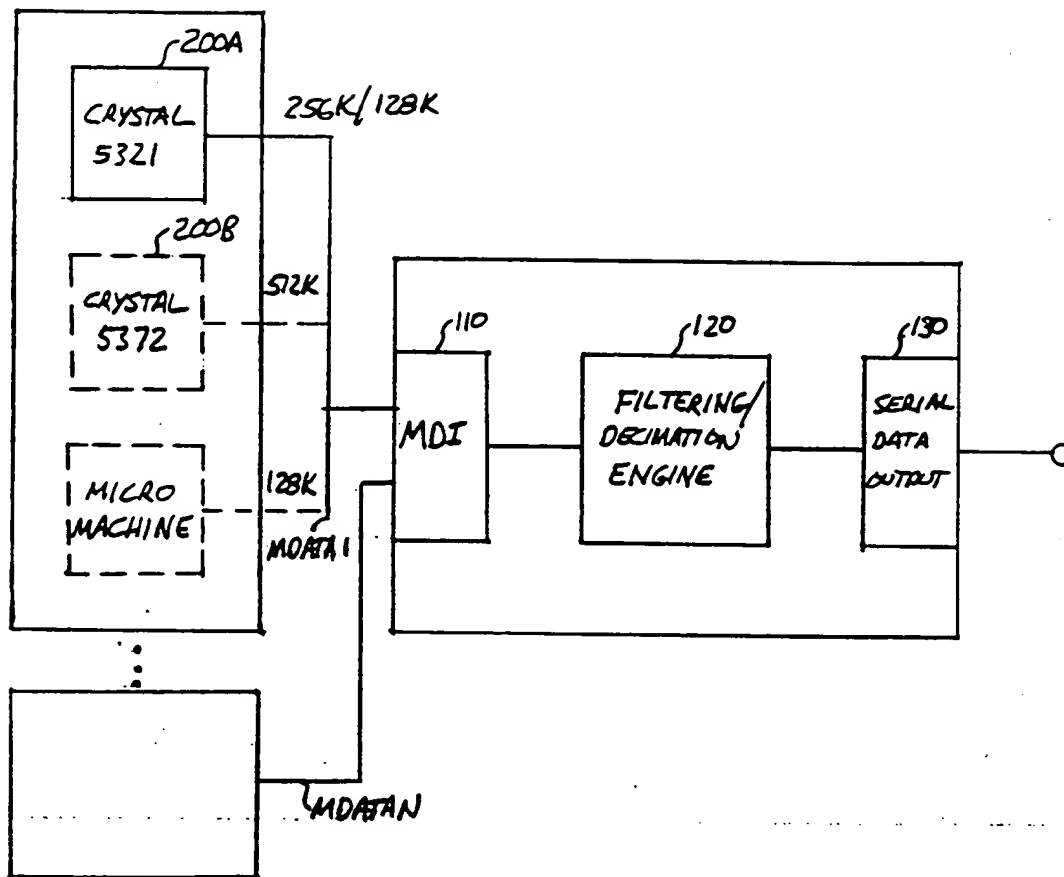


FIGURE 2

SELECTABLE VOLTAGE
EG. 2.5V V_{DD}
DIGITAL SUPPLY PIN

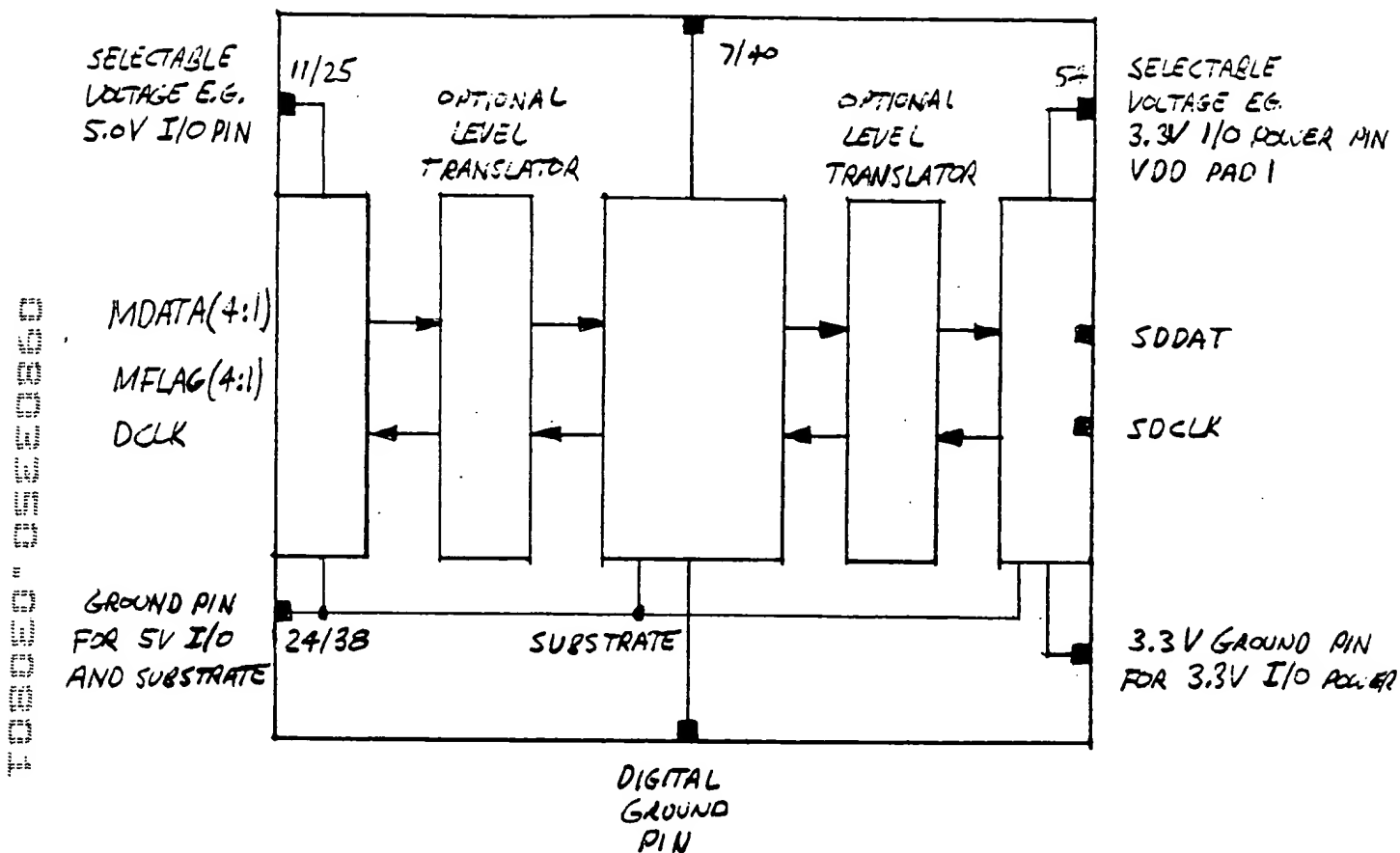


FIGURE 3

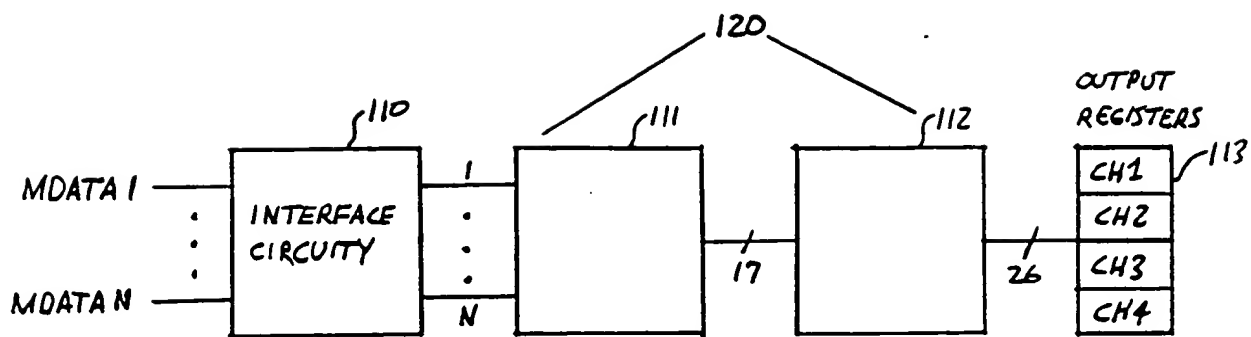


FIGURE 4

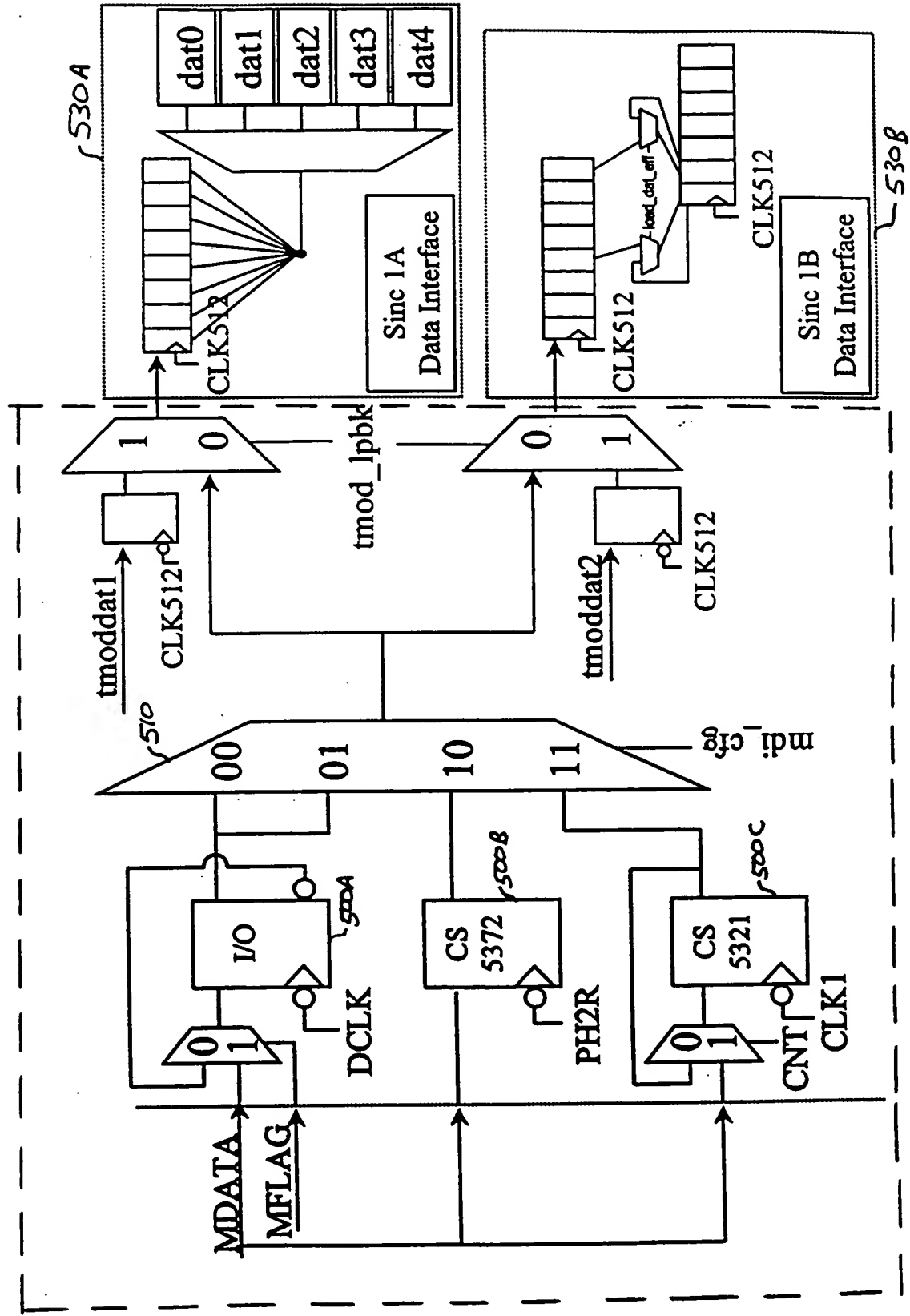


FIGURE 5

2. Bismarck Sinc Decimation Chain

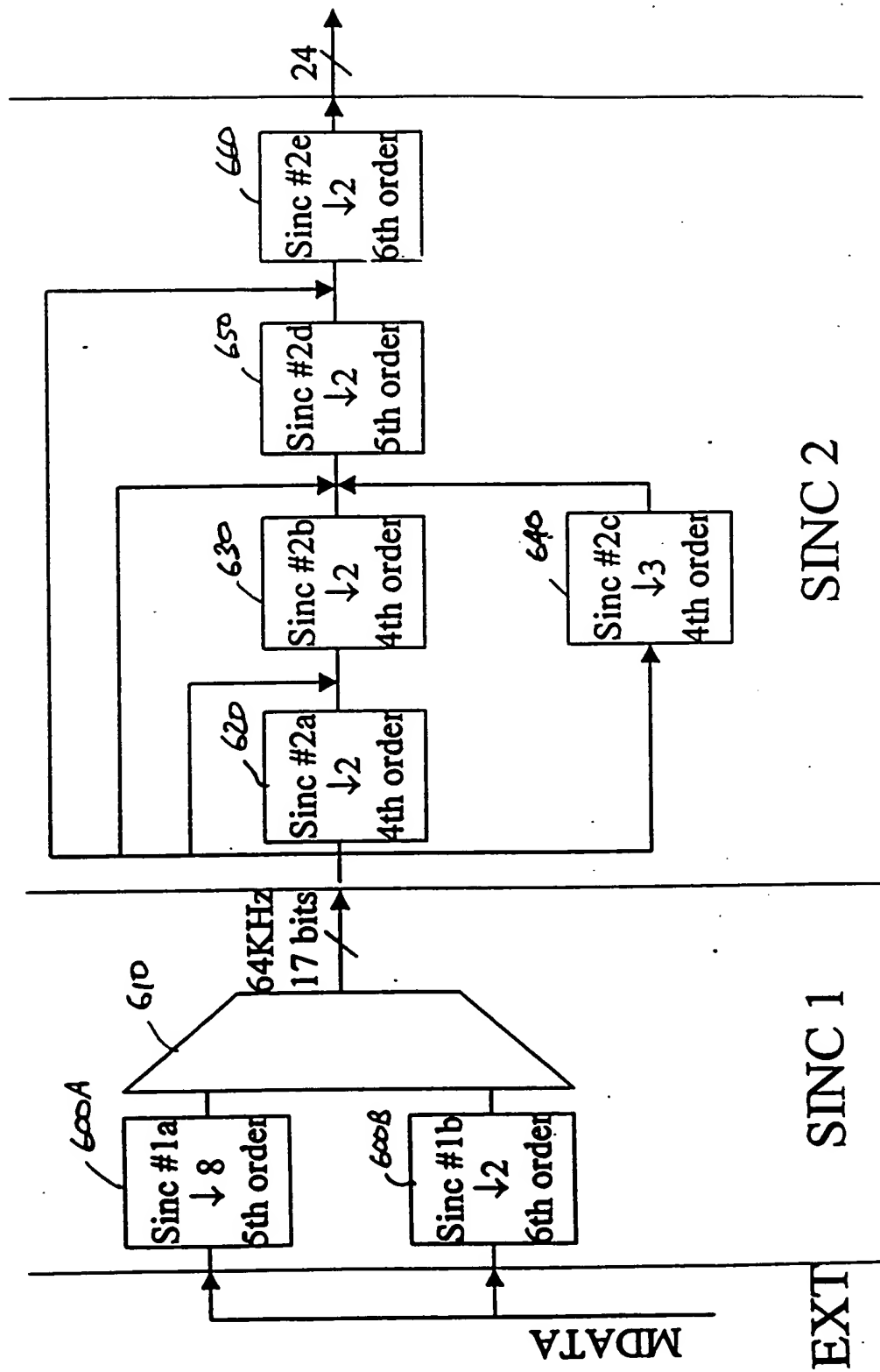


FIGURE 6

- Fifth order decimate by 8:

$$H(z) = \left(\frac{1-z^{-8}}{1-z^{-1}} \right)^5$$

- 36 tap FIR filter. Half of the (symmetric) coefficients

$h_0=1$	$h_1=5$	$h_2=15$	$h_3=35$	$h_4=70$	$h_5=126$	$h_6=210$	$h_7=330$	$h_8=490$
$h_9=690$	$h_{10}=926$	$h_{11}=1190$	$h_{12}=1470$	$h_{13}=1750$	$h_{14}=2010$	$h_{15}=2226$	$h_{16}=2380$	$h_{17}=2460$

FIGURE 7

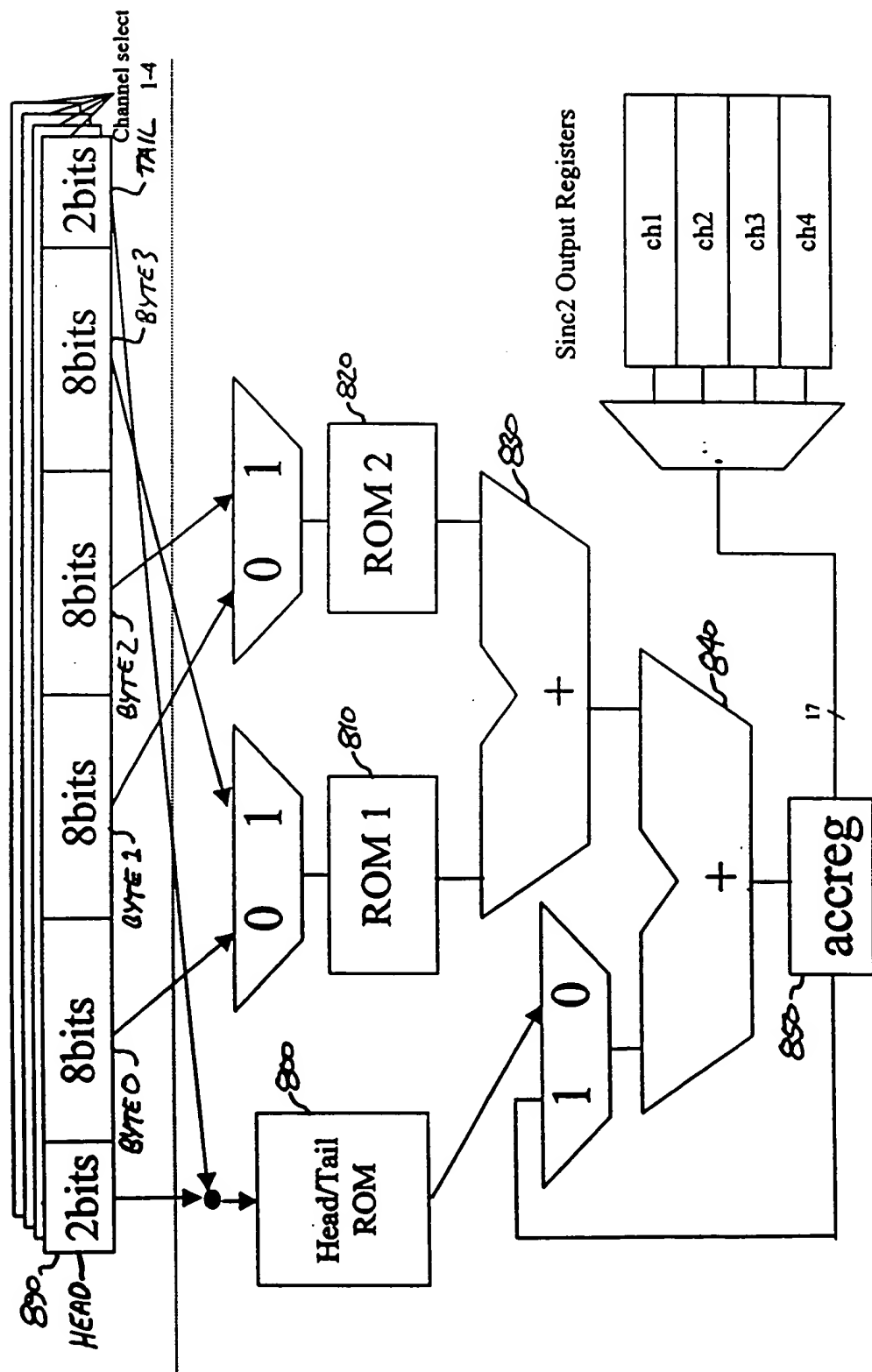


FIGURE 8

$$H(z) = \left(\frac{1-z^{-3}}{1-z^{-1}} \right)^4$$

Impulse Response:

$$y[n] = x[n] + 6 \cdot x[n-1] + 15 \cdot x[n-2] + 20 \cdot x[n-3] + 15 \cdot x[n-4] + 6 \cdot x[n-5] + x[n-6]$$

FIGURE 9

3. Bismarck Sinc1b Functional Diagram

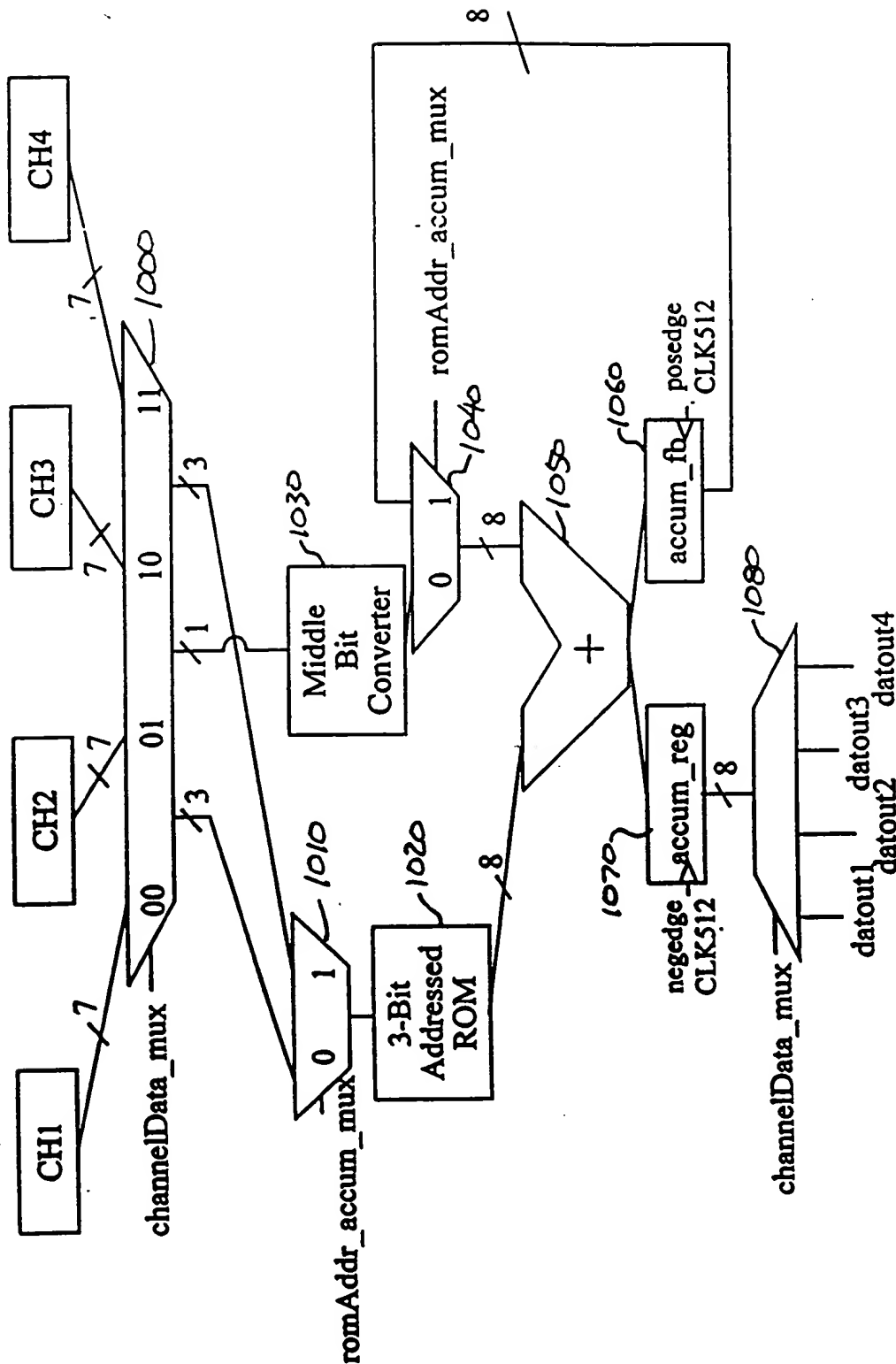


FIGURE 10

Filter Name	System Function	IMPULSE RESPONSE (FILTER COEFFICIENTS)
Sinc2(a) Sinc2(b)	$H(z) = \left(\frac{1-z^{-2}}{1-z^{-1}} \right)^4$	$h[n] = [1\ 4\ 6\ 4\ 1]$
Sinc2(c)	$H(z) = \left(\frac{1-z^{-3}}{1-z^{-1}} \right)^4$	$h[n] = [1\ 4\ 10\ 16\ 19\ 16\ 10\ 4\ 1]$
Sinc2(d)	$H(z) = \left(\frac{1-z^{-2}}{1-z^{-1}} \right)^3$	$h[n] = [1\ 5\ 10\ 10\ 5\ 1]$
Sinc2(e)	$H(z) = \left(\frac{1-z^{-2}}{1-z^{-1}} \right)^6$	$h[n] = [1\ 6\ 15\ 20\ 15\ 6\ 1]$

FIGURE 11

Sinc2(a) and Sinc2(b):

$$\begin{aligned} y[n] &= x[n] + 4x[n-1] + 6x[n-2] + 4x[n-3] + x[n-4] \\ &= x[n] + 4x[n-1] + 4x[n-2] + 2x[n-2] + 4x[n-3] + x[n-4] \end{aligned}$$

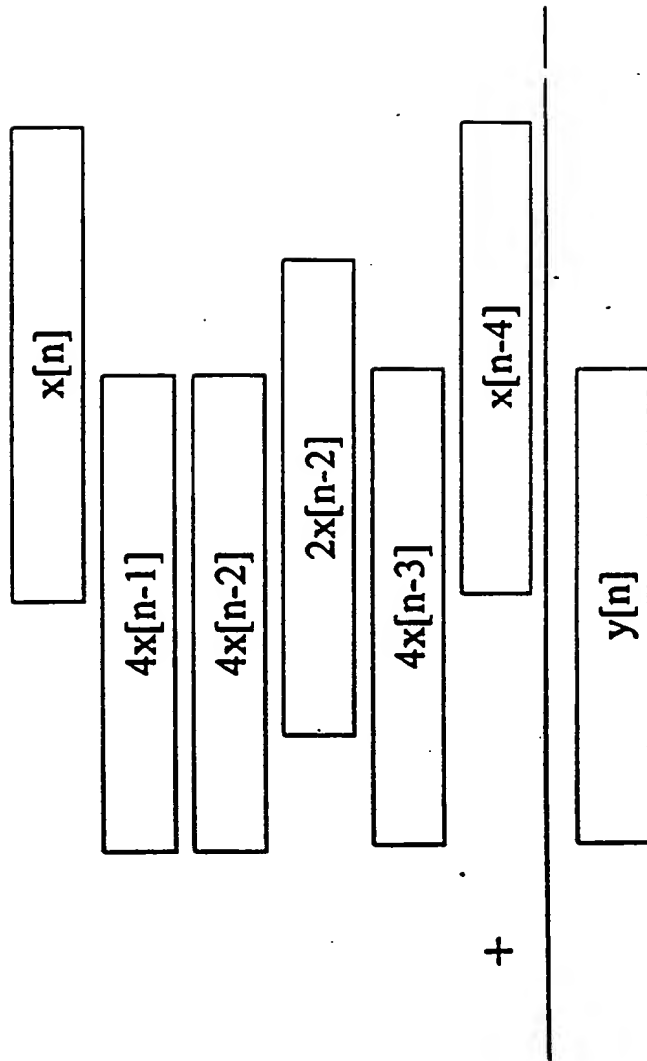


FIGURE 12

Figure 13A Sinc2(c):

$$\begin{aligned}
 y[n] &= x[n] + 4x[n-1] + 10x[n-2] + 16x[n-3] + 19x[n-4] + 16x[n-5] + 10x[n-6] + 4x[n-7] + x[n-8] \\
 &= x[n] + 4x[n-1] + 8x[n-2] + 2x[n-2] + 16x[n-3] + 16x[n-4] + 2x[n-4] + x[n-4] \\
 &\quad + 16x[n-5] + 8x[n-6] + 2x[n-6] + 4x[n-7] + x[n-8]
 \end{aligned}$$

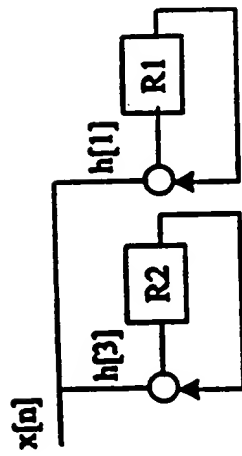
Figure 13B Sinc2(d):

$$\begin{aligned}
 y[n] &= x[n] + 5x[n-1] + 10x[n-2] + 10x[n-3] + 5x[n-4] + x[n-5] \\
 &= x[n] + 4x[n-1] + x[n-1] + 8x[n-2] + 2x[n-2] + 8x[n-3] + 2x[n-3] + 4x[n-4] + x[n-5]
 \end{aligned}$$

Figure 13C Sinc2(e):

$$\begin{aligned}
 y[n] &= x[n] + 6x[n-1] + 15x[n-2] + 20x[n-3] + 15x[n-4] + 6x[n-5] + x[n-6] \\
 &= x[n] + 4x[n-1] + 2x[n-1] + 16x[n-2] - x[n-2] + 16x[n-3] + 4x[n-3] \\
 &\quad + 16x[n-4] - x[n-4] + 4x[n-5] + 2x[n-5] + x[n-6]
 \end{aligned}$$

Sinc2(a) and Sinc2(b):

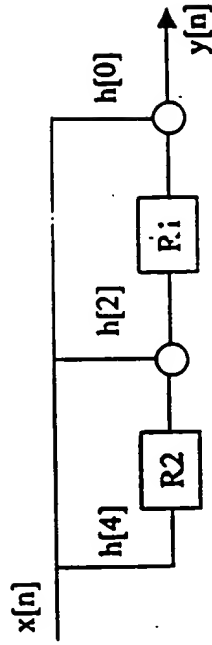


Accumulate Phase (2 additions)

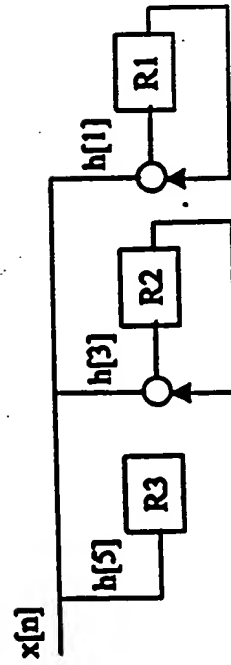
FIGURE 14A

Output Phase (4 additions)

FIGURE 14B



Sinc2(d):

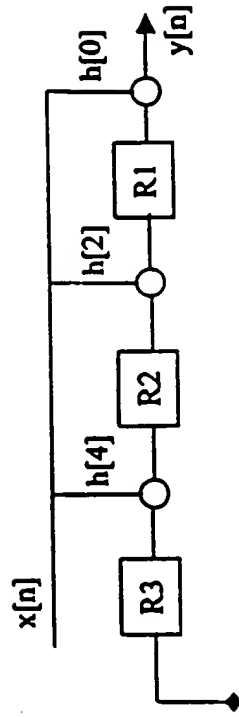


Accumulate Phase (5 additions)

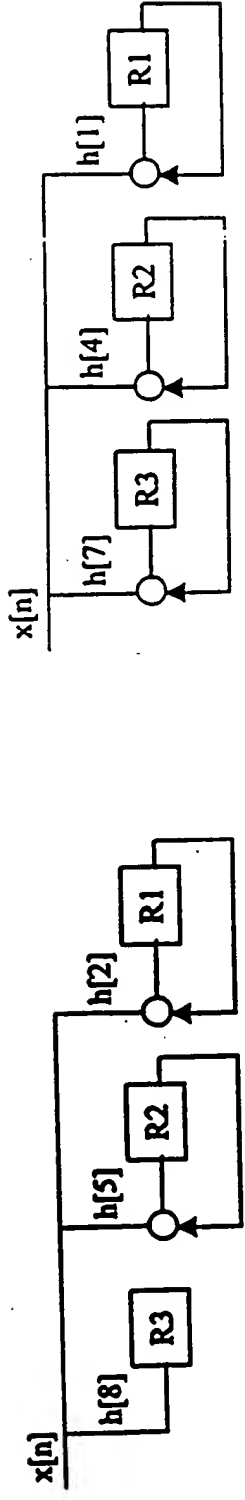
FIGURE 15A

Output Phase (5 additions)

FIGURE 15B



Sinc2(c):

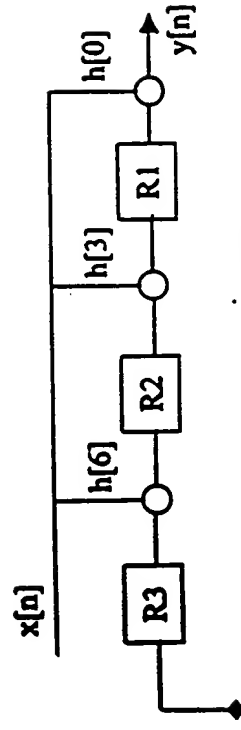


Accumulate Phase A (4 additions)

FIGURE 16A

Accumulate Phase B (5 additions)

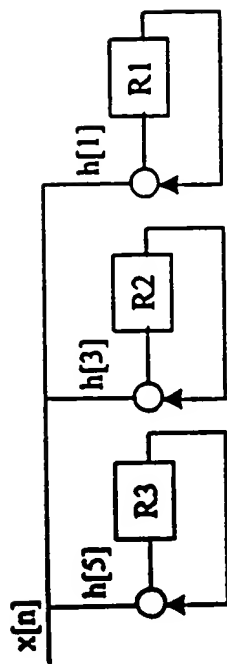
FIGURE 16B



Output Phase (4 additions)

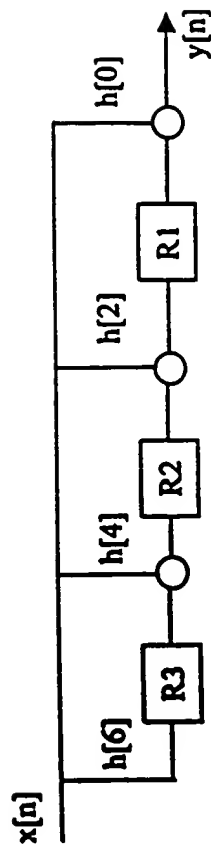
FIGURE 16C

Sinc2(e):



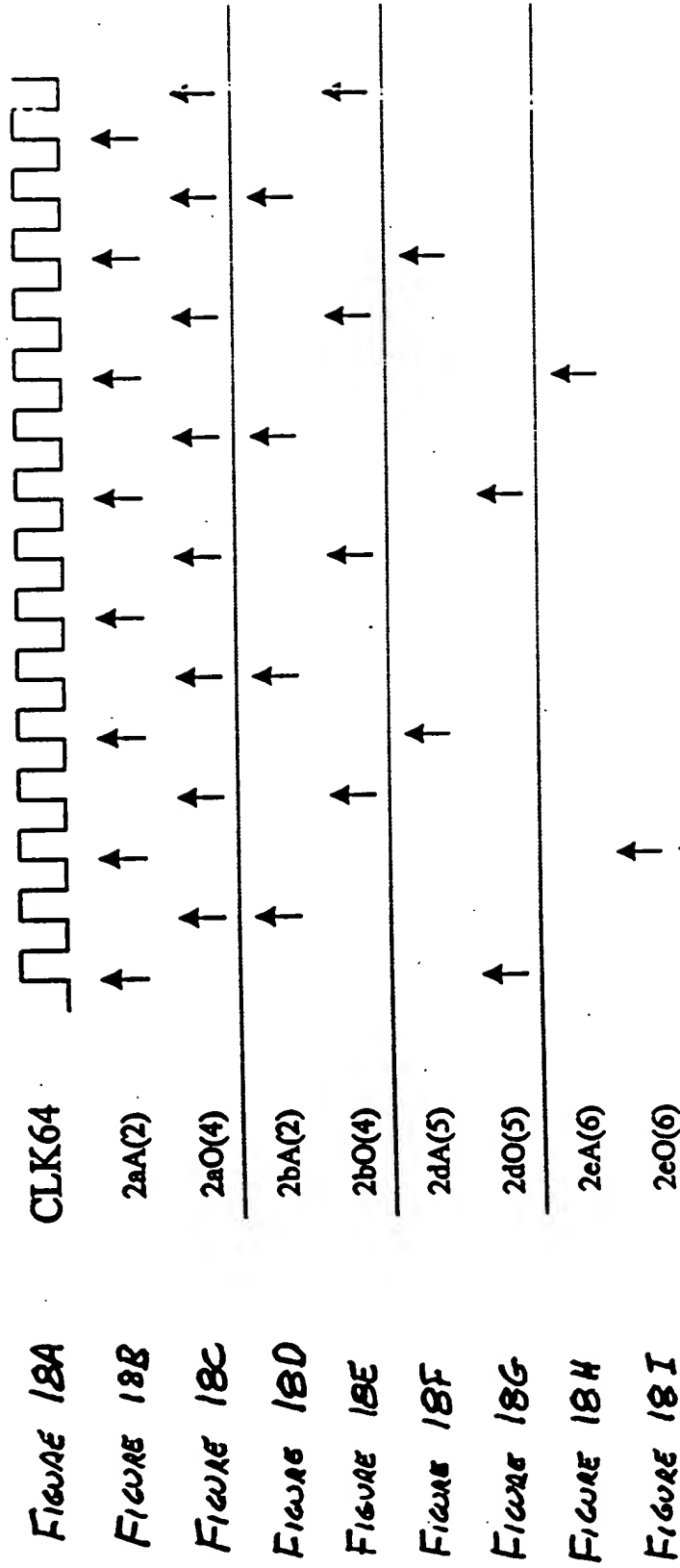
Accumulate Phase (6 additions)

FIGURE 17A



Output Phase (6 additions)

FIGURE 17B



RAM1

Main Program

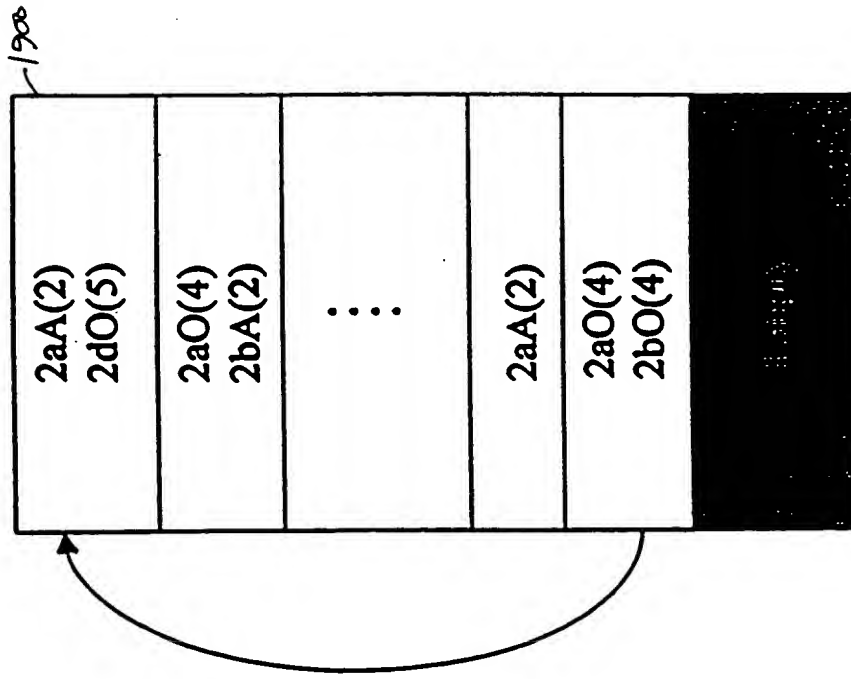


FIGURE 19A

RAM2

Subroutines

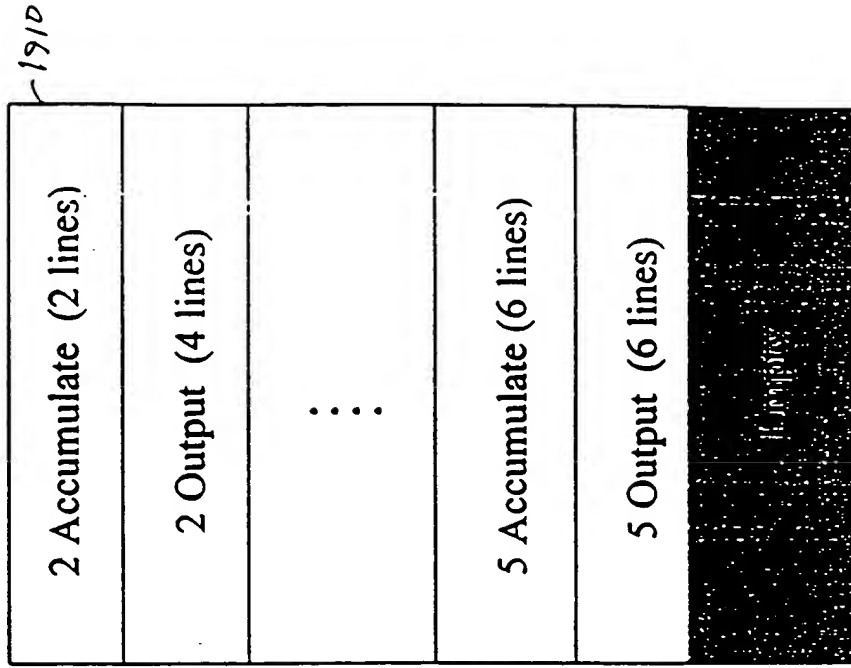


FIGURE 19B

Sinc2 Control-Datapath Architecture

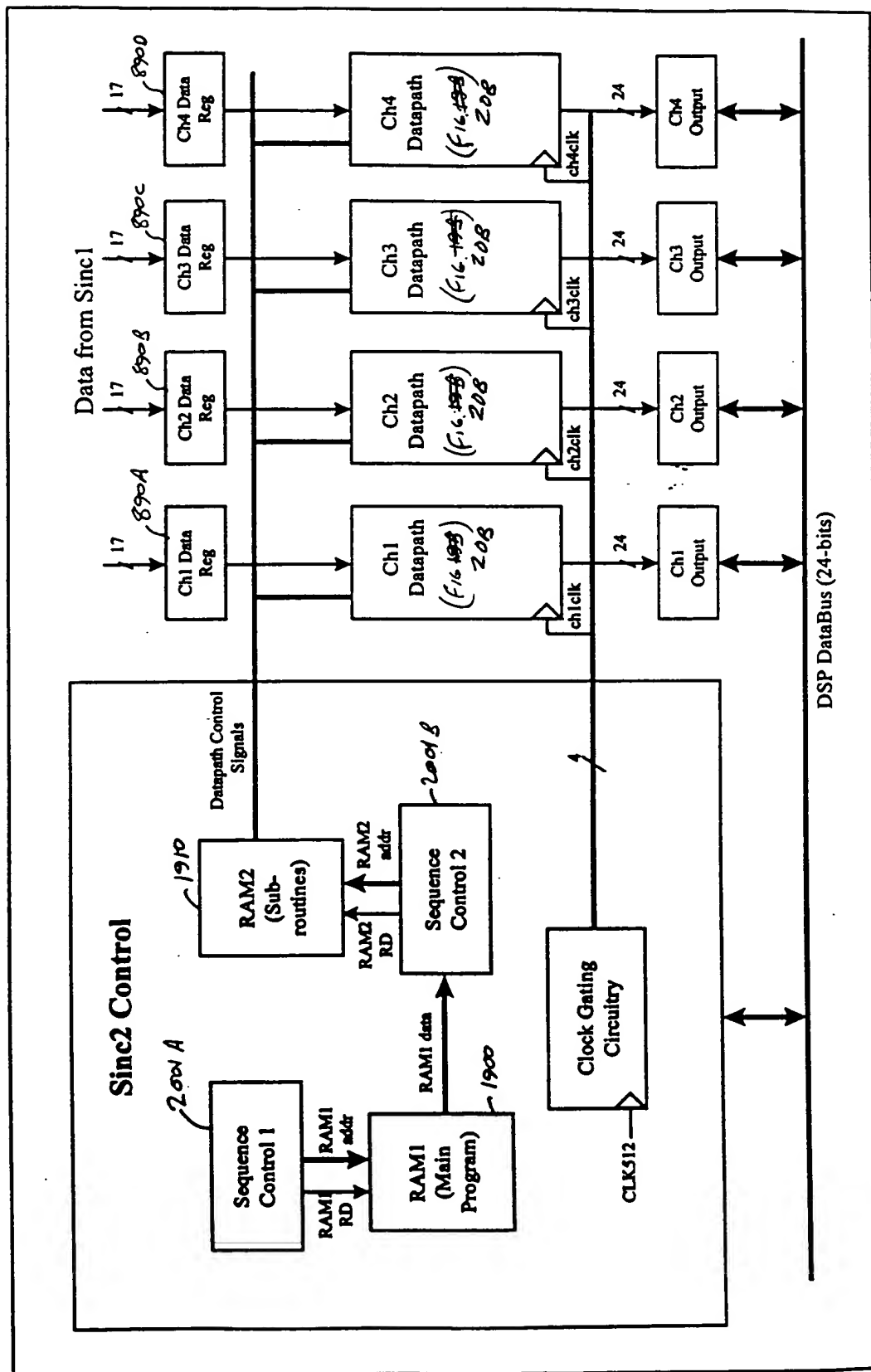


FIGURE 20A

FIGURE 20B

Programming Procedure:

1. Select decimation rate.
2. Select required mini-sincs and associated Accumulate and Output subroutines.
3. Separate coefficients into form suitable for shift-add operations.
4. Check for overflow after each addition in the filter.
5. Perform necessary truncation to 24 bits and scaling of subsequent coefficients in mini-sincs.
6. Time multiplex Accumulate and Output Subroutines so that a maximum of 8 additions/subtractions are performed for each input from sinc1.
7. Create code for RAM2 (Accumulate and Output Subroutines) in the form:
 [Coeff 1] [Src 1] [Src 2] [Dest] [Coeff2] [Done Subroutine]
8. Create code for RAM1 (Main Control code)
 [Line #] [Wait for new data] [Done program]

FIGURE 21

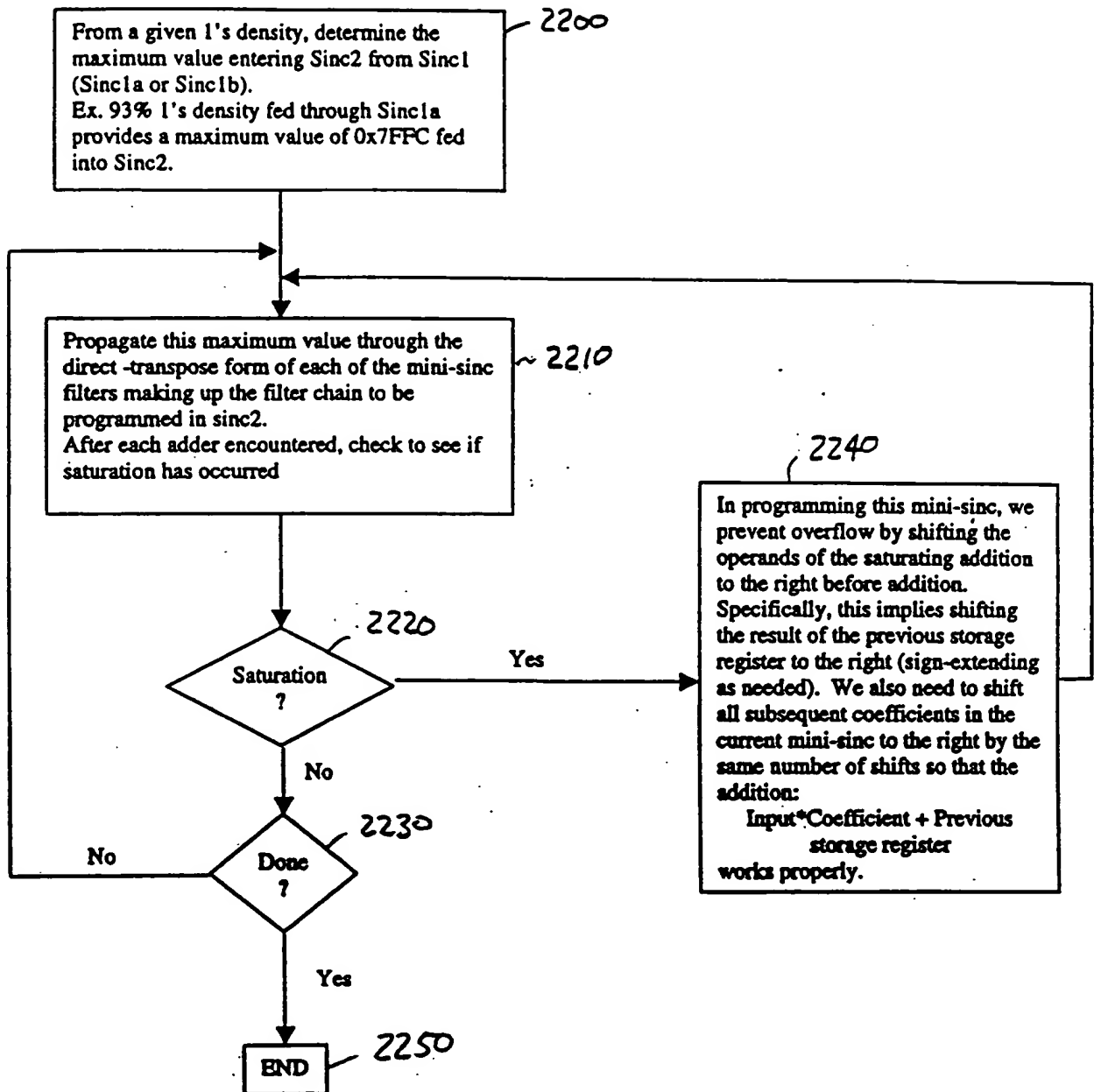


FIGURE 22

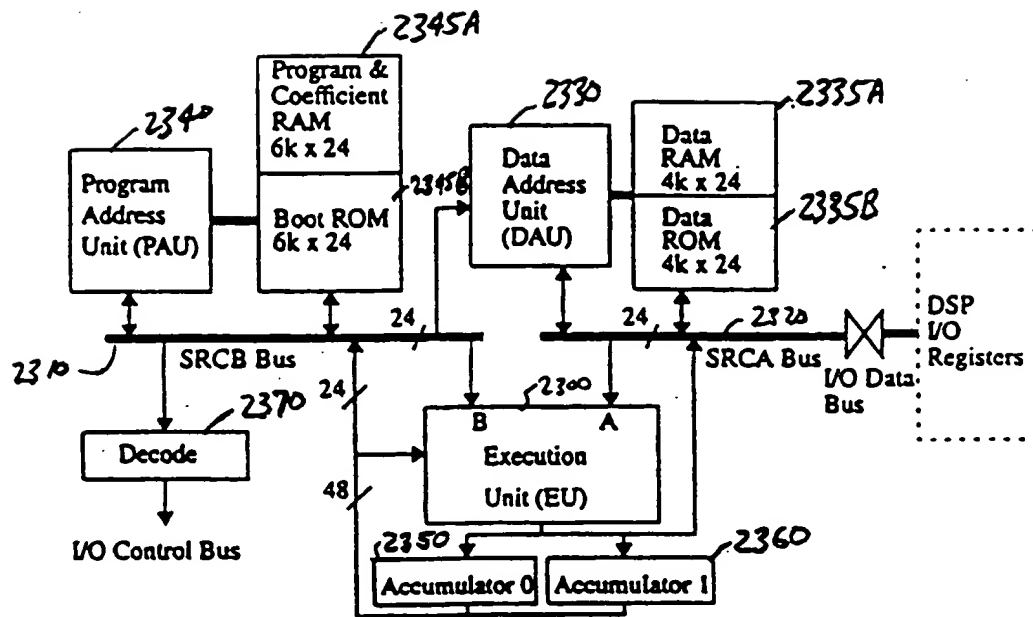


FIGURE 23

DSP Program Memory Map

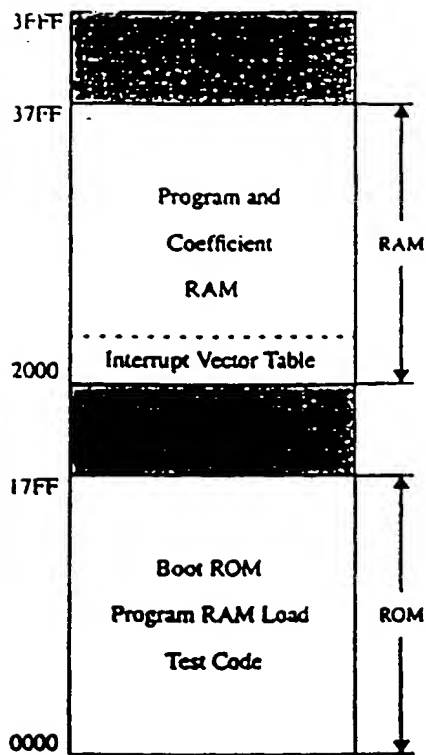


FIGURE 24A

DSP Data Memory/Register Map

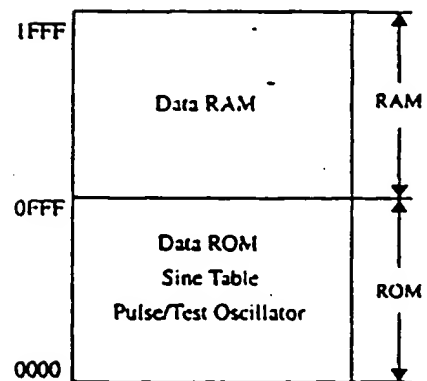


FIGURE 24B

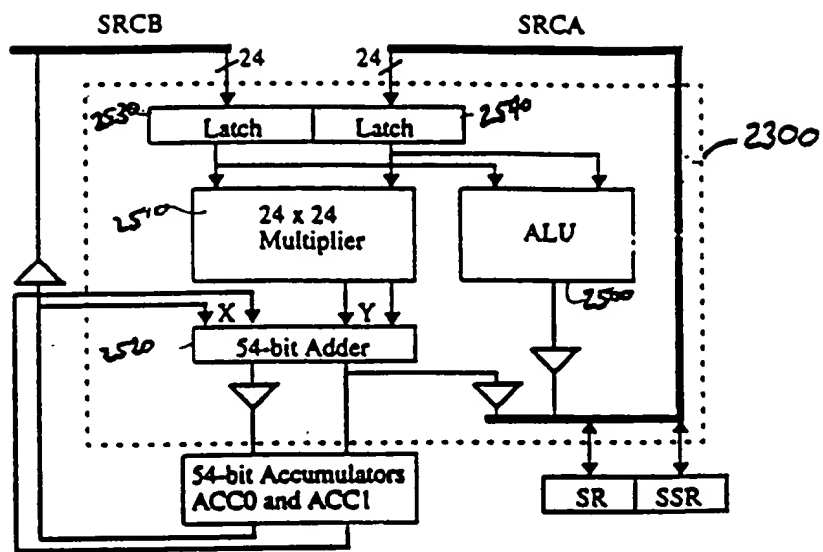


FIGURE 25

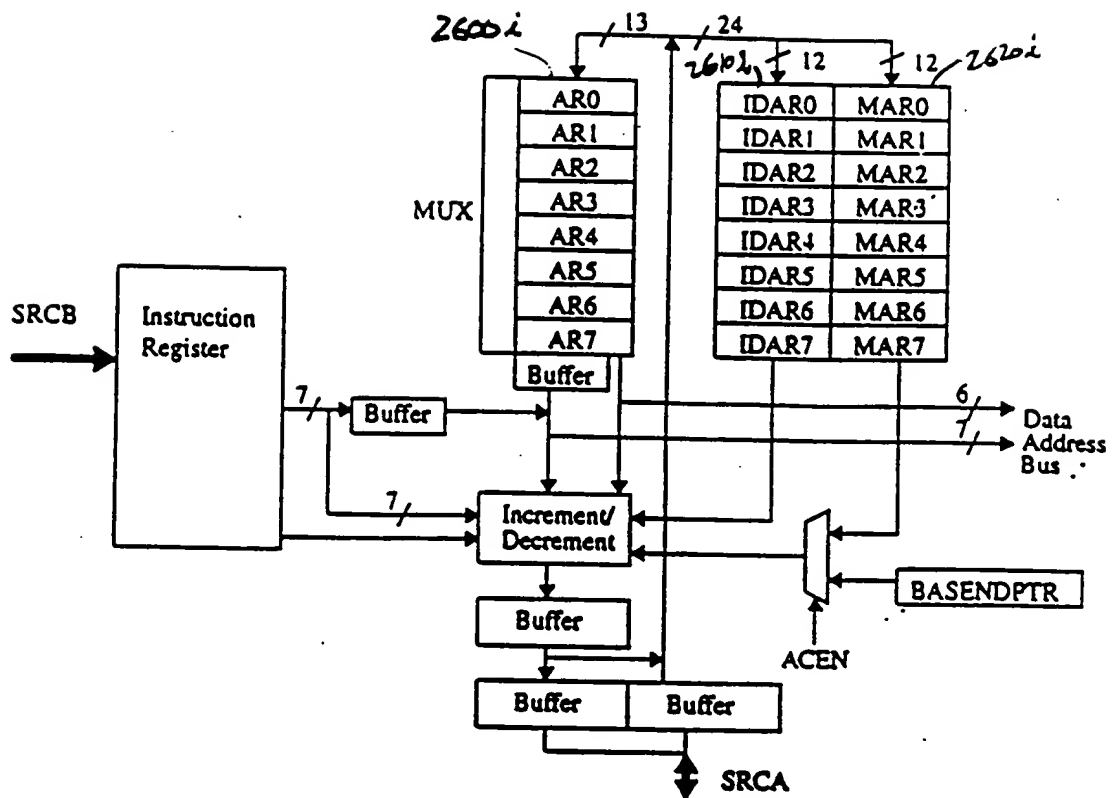


FIGURE 26

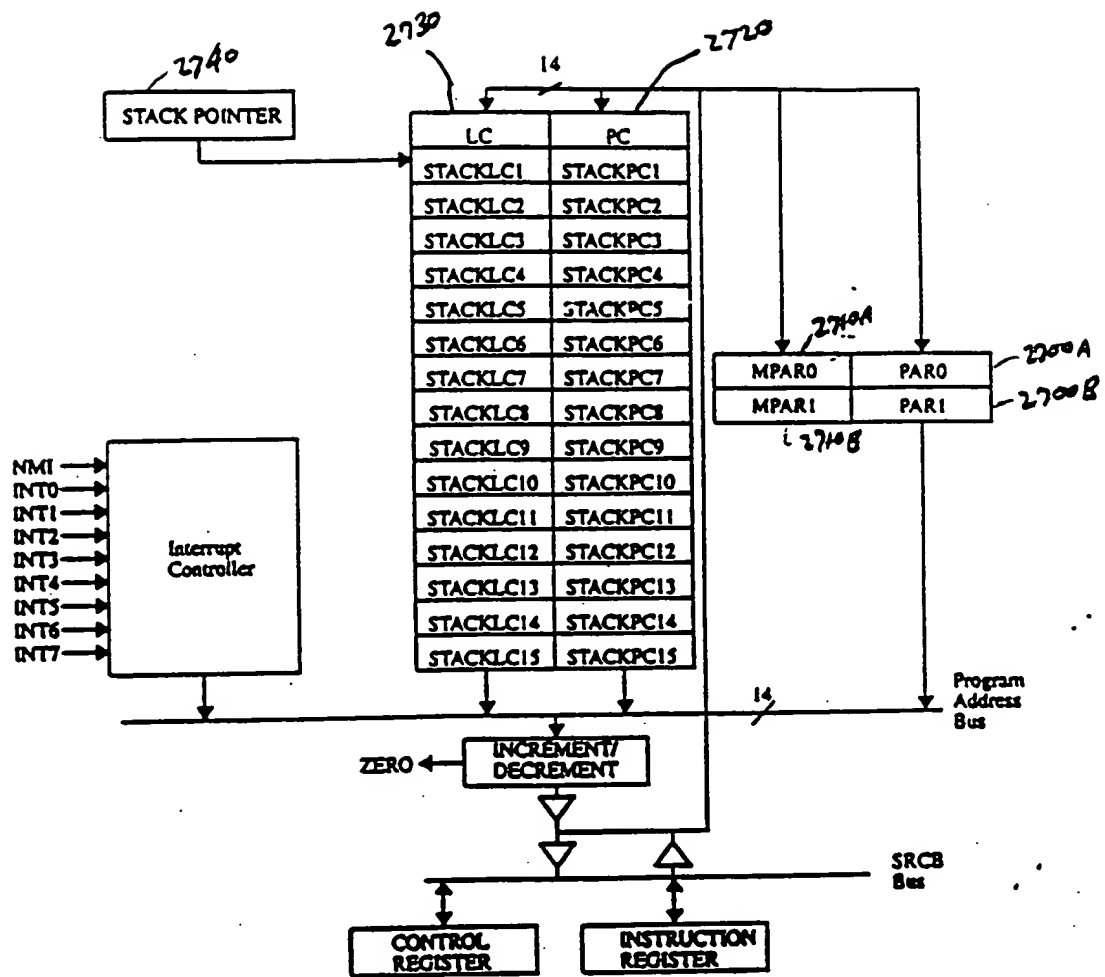


FIGURE 27

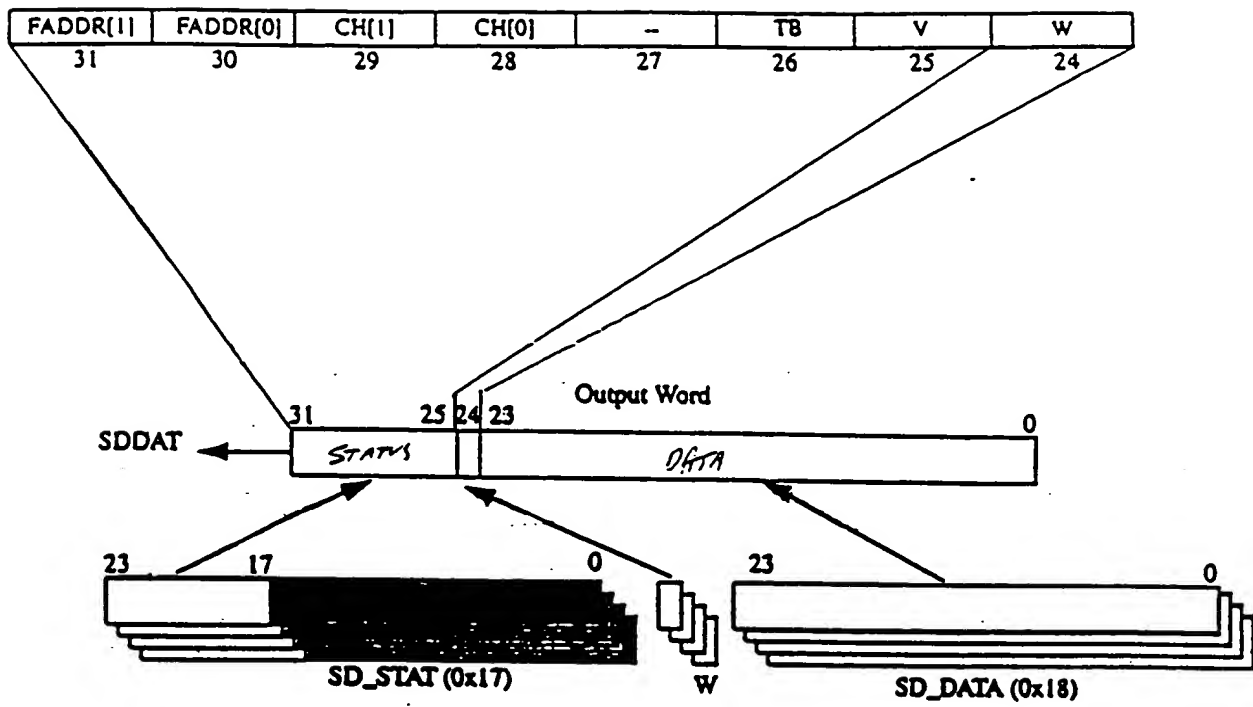


FIGURE 28

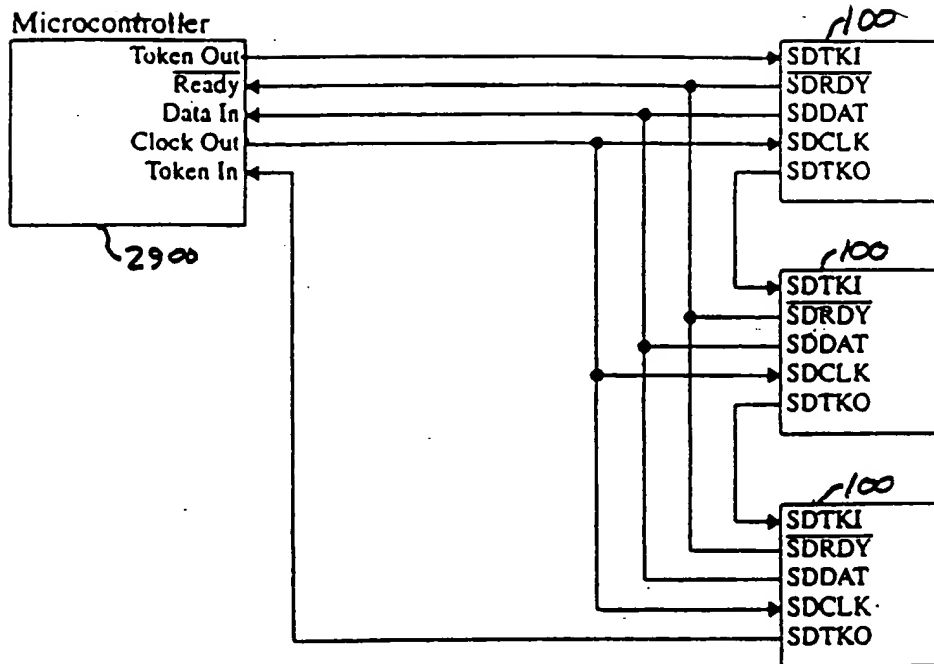


FIGURE 29

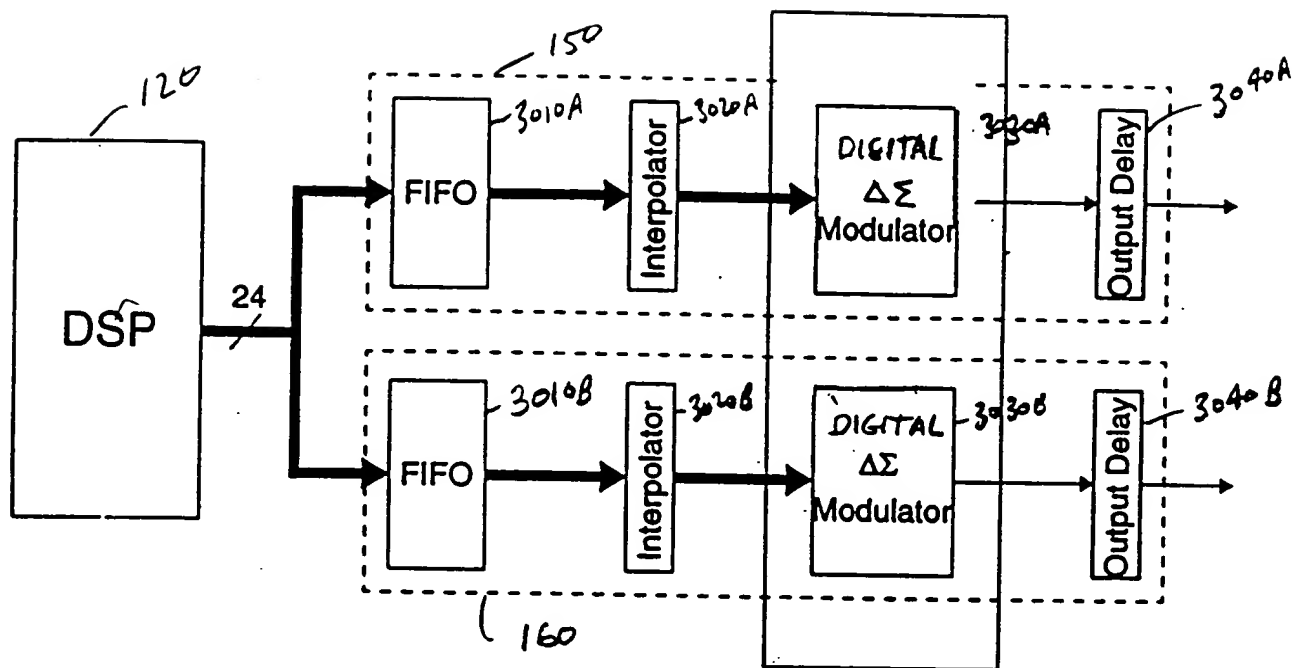


FIGURE 30A

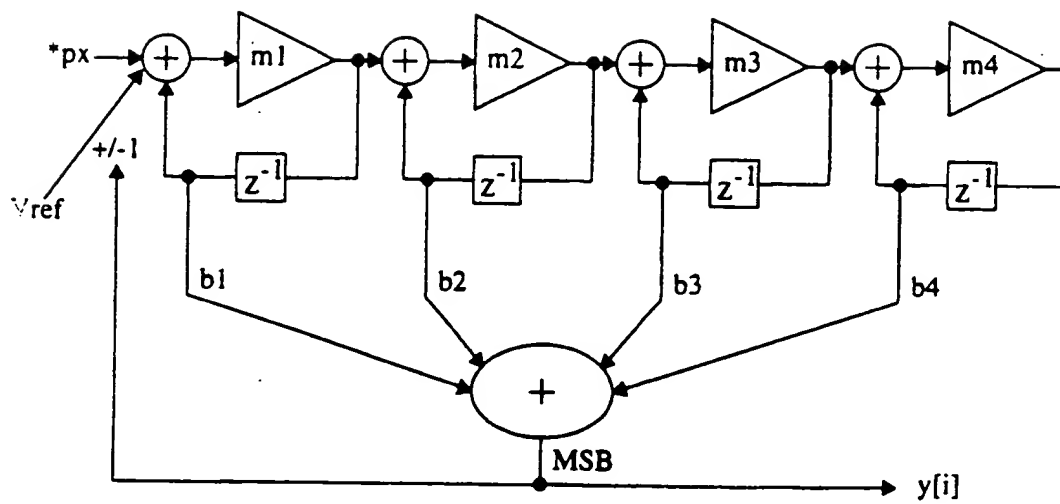


FIGURE 30B

FIGURE 30C1

 wire

FIGURE 30C2

 24 wires

FIGURE 30C3

 register

FIGURE 30C4


 multiplexer

FIGURE 30C5

 tristate buffer

FIGURE 30C6


 inverter

FIGURE 30C7

 exclusive or gate

FIGURE 30C8

 adder

FIGURE 30C9

 multiplier

FIGURE 30C10

 right shifter

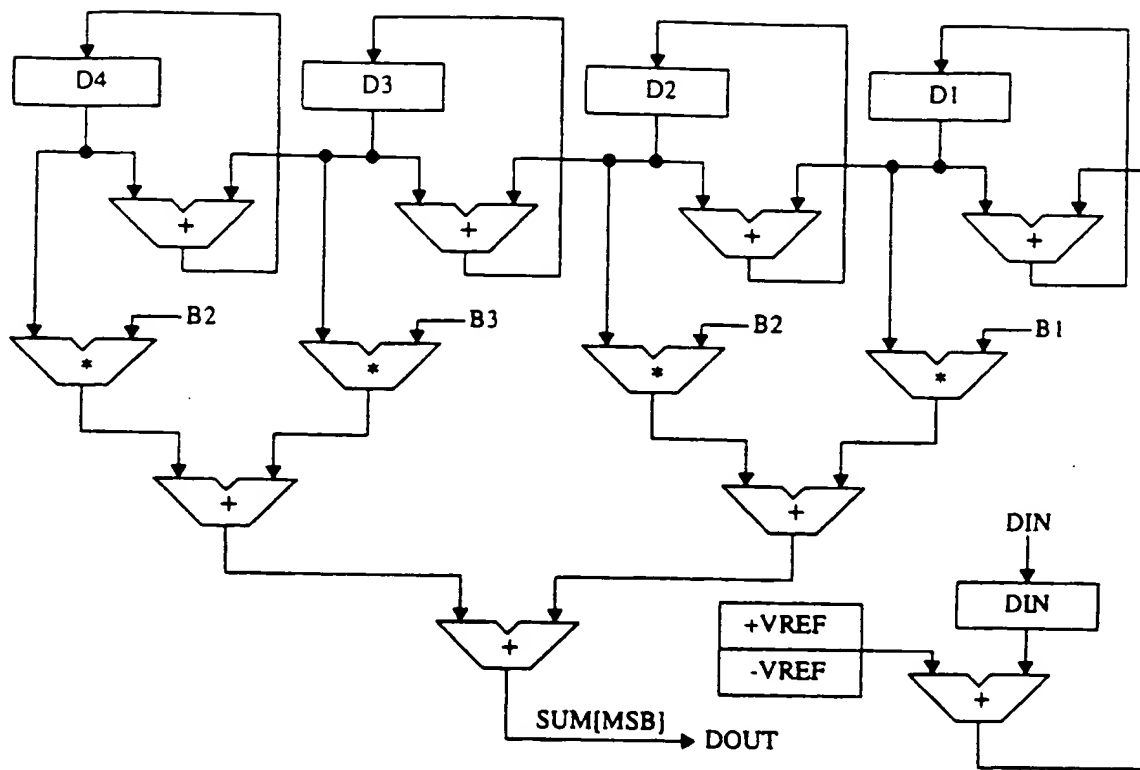


FIGURE 30D

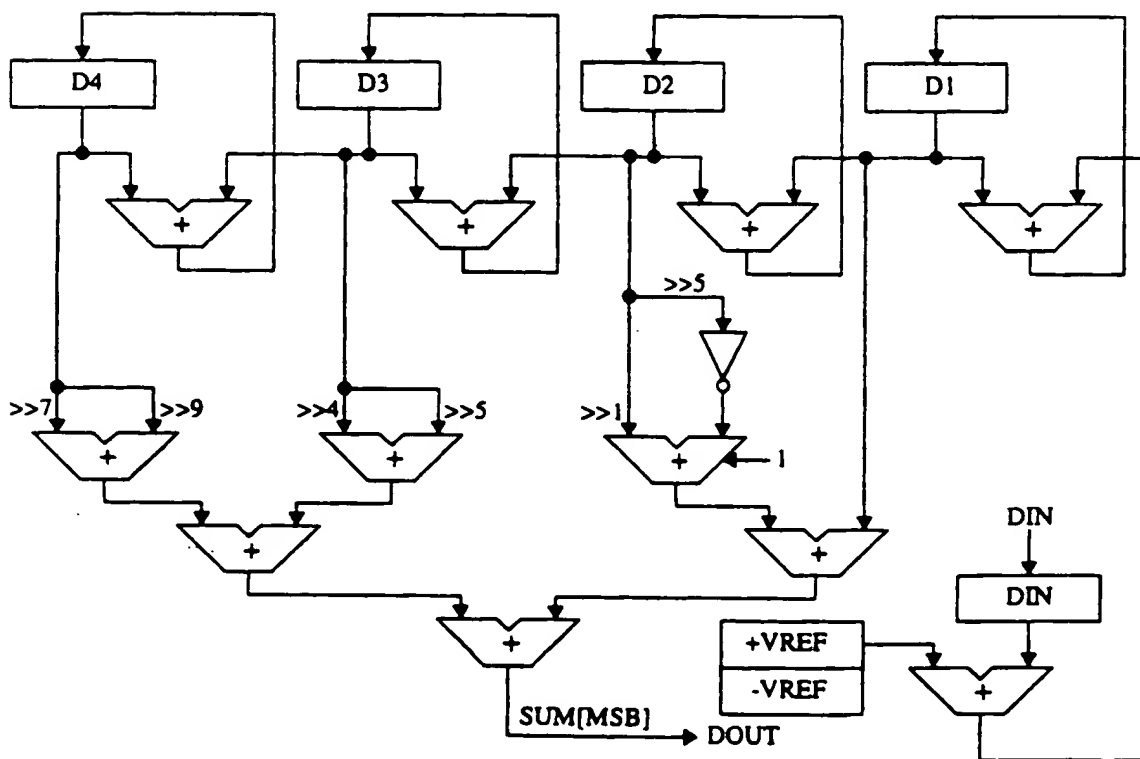


FIGURE 30E

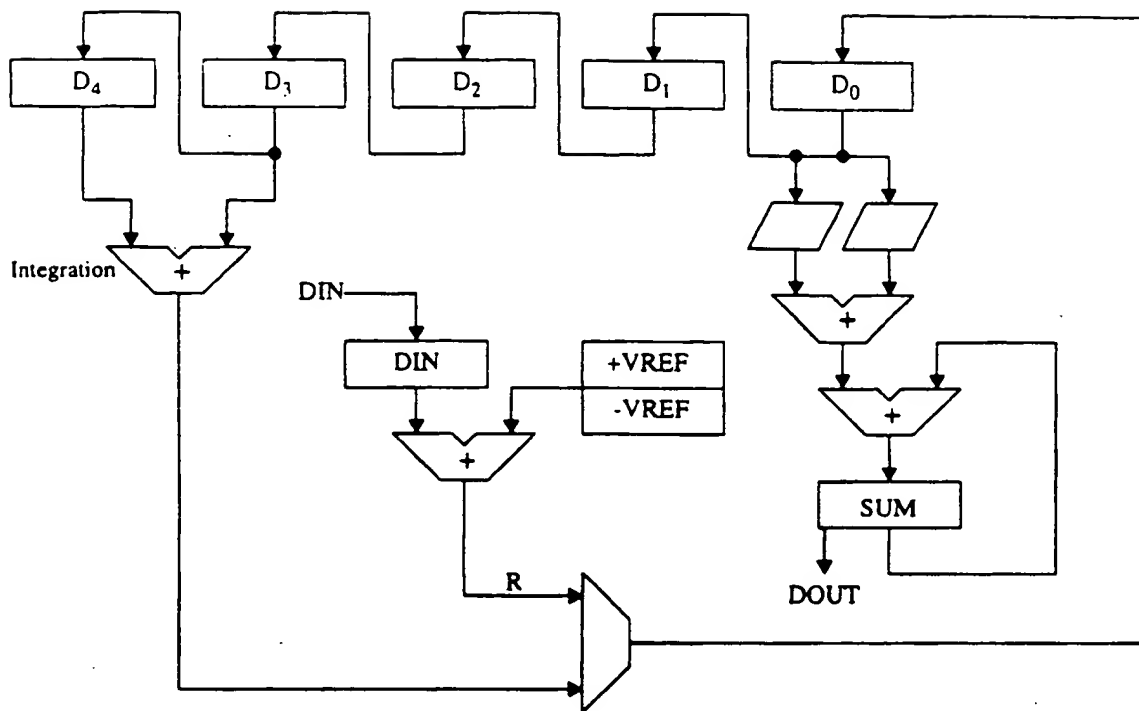


FIGURE 30F1

State	Actions During State		
S0	$D_0(D4_k) = D_4(D4_{k-1}) + D_3(D3_{k-1})$	Clear SUM	Load DIN_k
S1	$D_0(D3_k) = D_4(D3_{k-1}) + D_3(D2_{k-1})$	$SUM_k += D_0(D4_k) \gg \text{Shift4}$	
S2	$D_0(D2_k) = D_4(D2_{k-1}) + D_3(D1_{k-1})$	$SUM_k += D_0(D3_k) \gg \text{Shift3}$	
S3	$D_0(D1_k) = D_4(D1_{k-1}) + D_3(R_{k-1})$	$SUM_k += D_0(D2_k) \gg \text{Shift2}$	
S4		$SUM_k += D_0(D1_k) \gg \text{Shift1}$	
S5	$D_0(R_k) = DIN_k +/- VREF$		

FIGURE 30F2

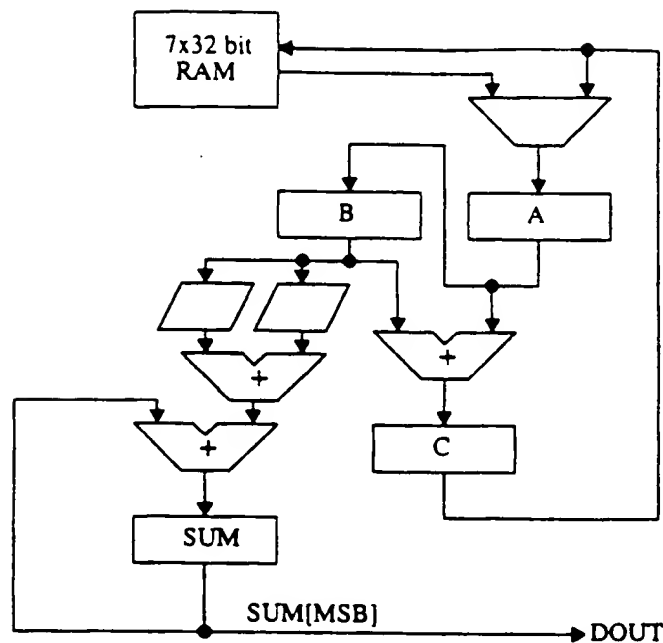


FIGURE 30 G1

State	Actions During State			
S0	Clear SUM	Clear C	Clear B	Clear A
S1				Load A<Mem(D4 _k)
S2			Shift B<A(D4 _k)	Load A<Mem(D3 _k)
S3	SUM _k += B(D4 _k)>>Shift4	C = B(D4 _k) + A(D3 _k)	Shift B<A(D3 _k)	Load A<Mem(D2 _k)
S4				Store C>Mem(D4 _{k+1})
S5	SUM _k += B(D3 _k)>>Shift3	C = B(D3 _k) + A(D2 _k)	Shift B<A(D2 _k)	Load A<Mem(D1 _k)
S6				Store C>Mem(D3 _{k+1})
S7	SUM _k += B(D2 _k)>>Shift2	C = B(D2 _k) + A(D1 _k)	Shift B<A(D1 _k)	Load A<Mem(DIN _k)
S8				Store C>Mem(D2 _{k+1})
S9	SUM _k += B(D1 _k)>>Shift1	C = B(D1 _k) + A(DIN _k)	Shift B<A(DIN _k)	Load A<Mem(VREF)
S10			Shift B<A(VREF)	LoadReg A<C(Temp)
S11		C = +/-B(VREF) + A(Temp)		
S12				Store C>Mem(D1 _{k+1})

FIGURE 30 62

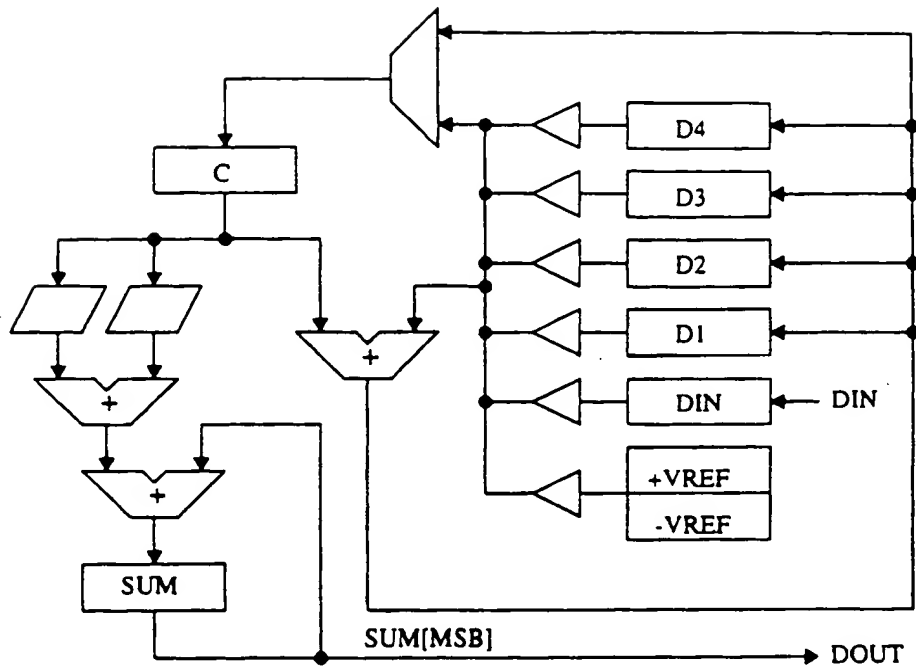


FIGURE 30 H1

State	Actions During State			
S0	Clear SUM	Load C < D4 _k		Load DIN _k
S1	SUM _k += C(D4 _k) >> Shift4	Load C < D3 _k	D4 _{k+1} = C(D4 _k) + D3 _k	
S2	SUM _k += C(D3 _k) >> Shift3	Load C < D2 _k	D3 _{k+1} = C(D3 _k) + D2 _k	
S3	SUM _k += C(D2 _k) >> Shift2	Load C < D1 _k	D2 _{k+1} = C(D2 _k) + D1 _k	
S4	SUM _k += C(D1 _k) >> Shift1	C(Temp) = C(D1 _k) + DIN _k		
S5			D1 _{k+1} = C(Temp) +/- VREF	

FIGURE 30 H2

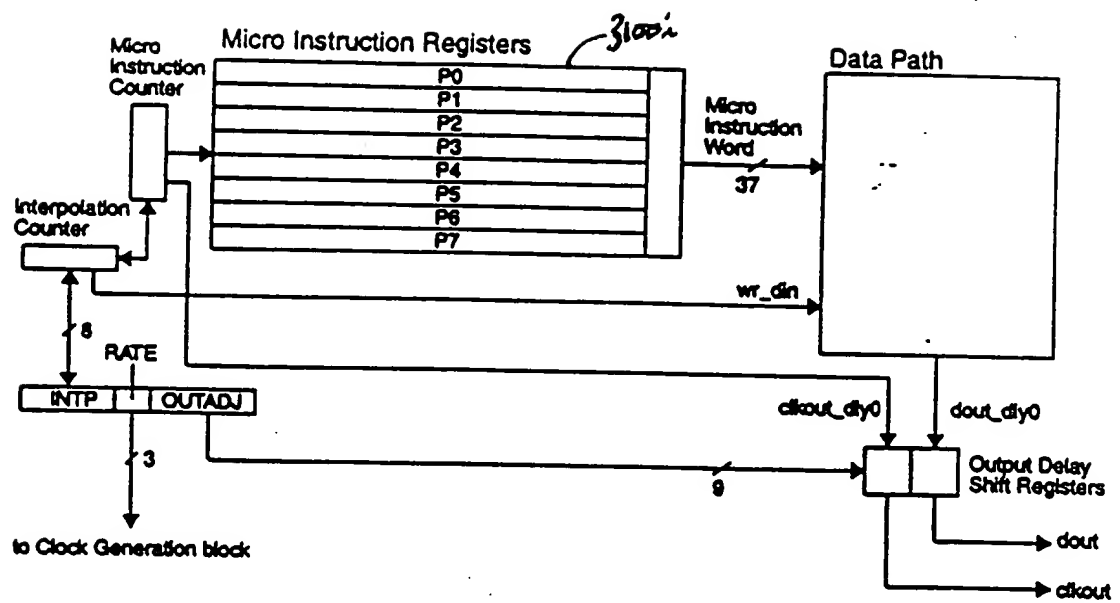


FIGURE 31

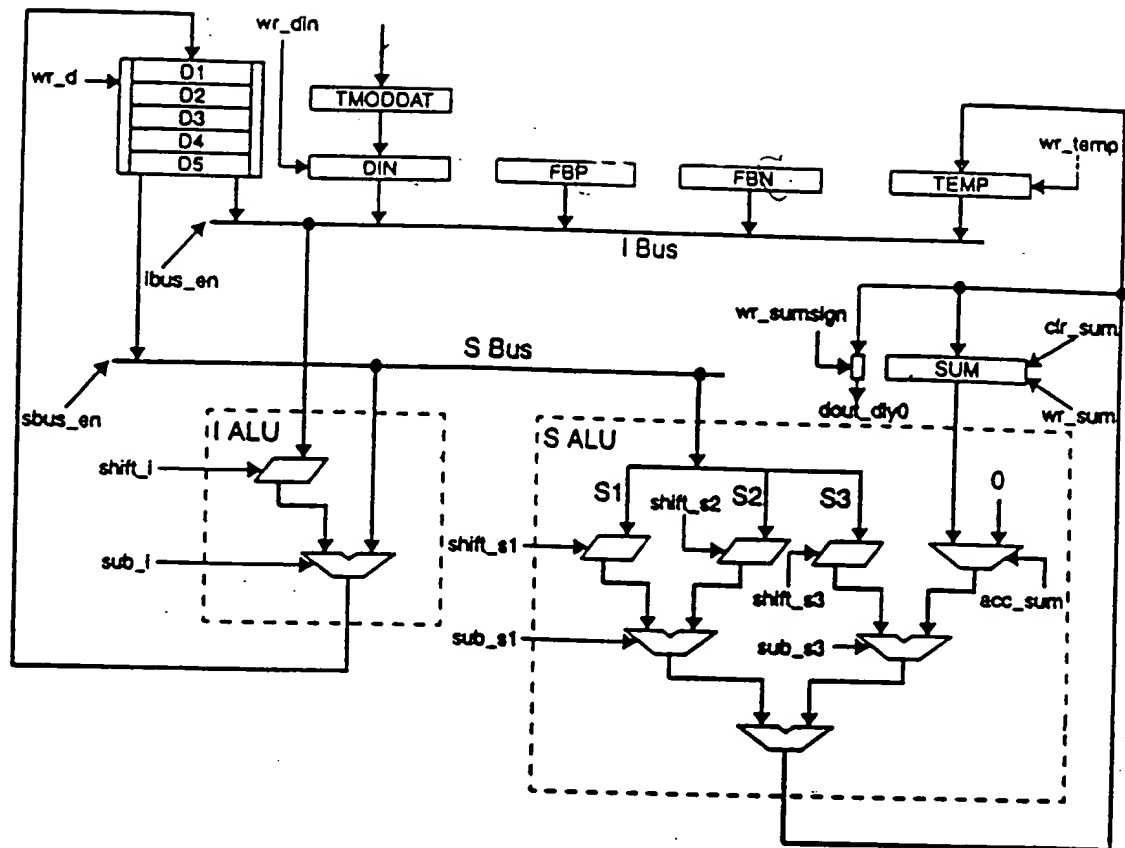


FIGURE 32

[illegible]

FIGURE 33

• 3

[illegible]

FIGURE 34

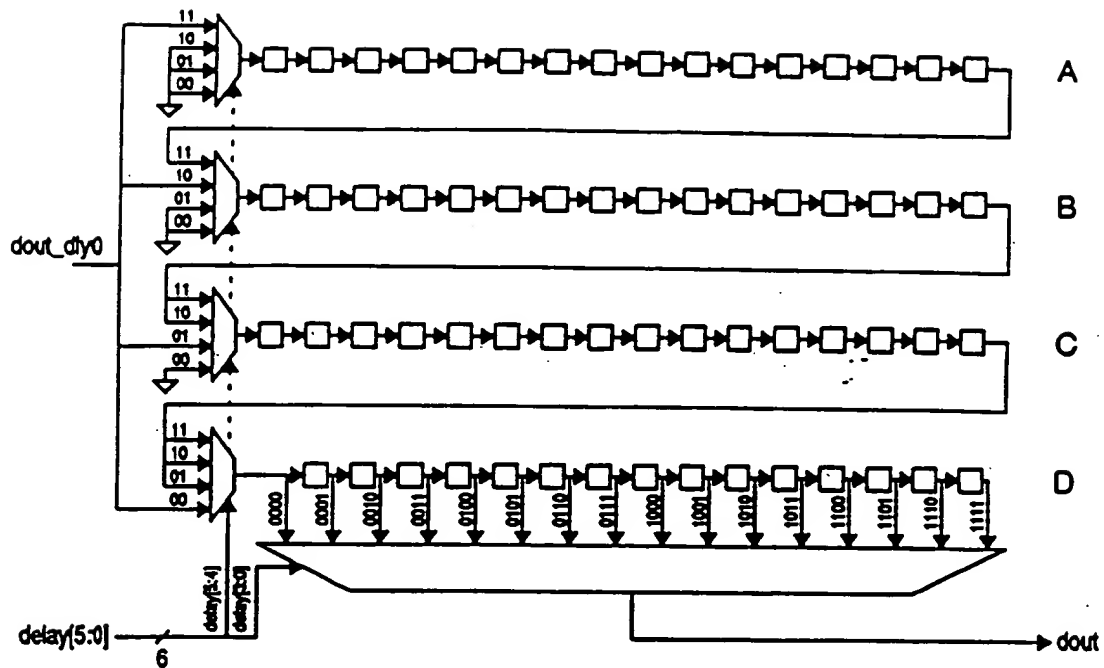


FIGURE 35A

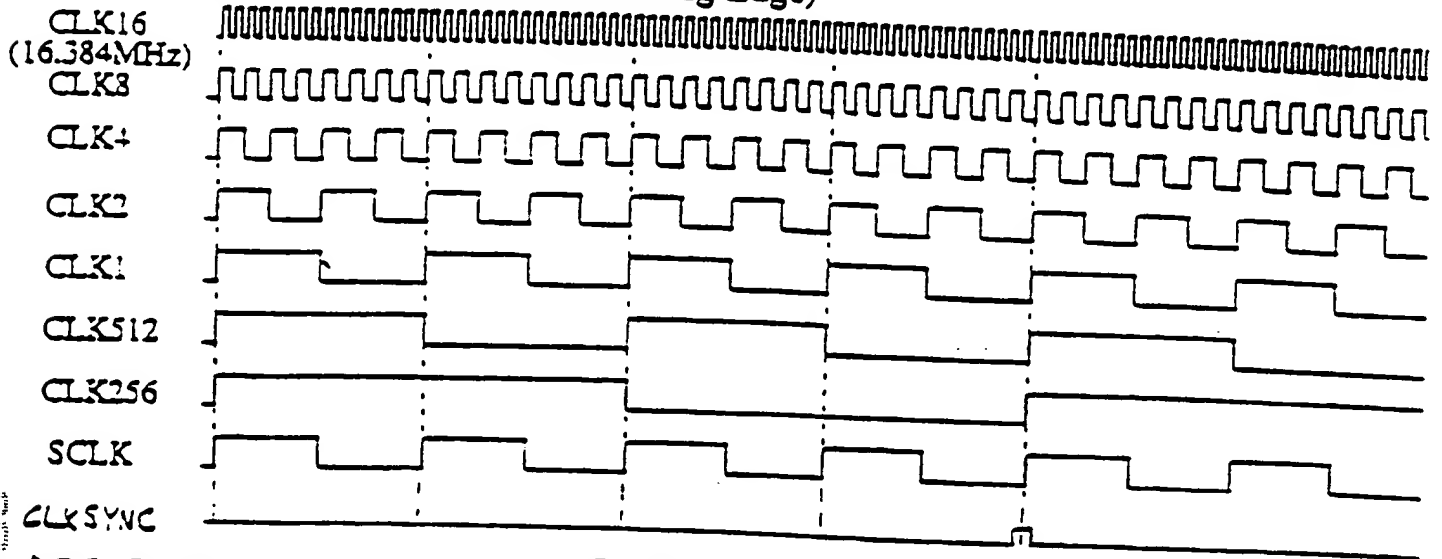
Table 1: Legend

dout_dly0	data output bit, 0 delay
dout	data output bit, 0-63 clock delay
delay[5:0]	how many clocks (0-63) to delay output data dout_dly0
delay[5:4]	selects segment into which to direct dout_dly0
delay[3:0]	selects where to tap segment D to get dout

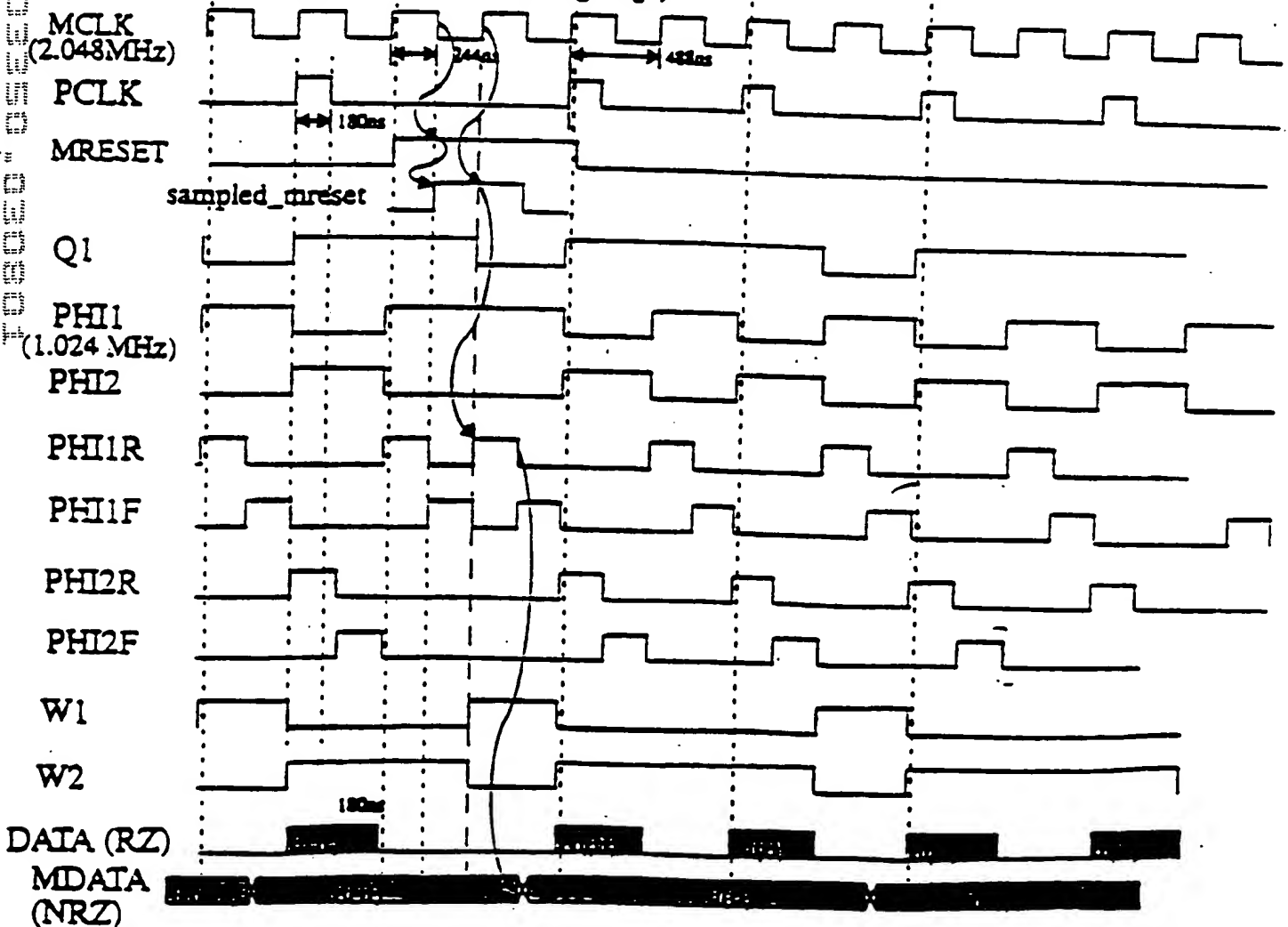
FIGURE 35B

RSU & ADC interface Clock Relationships with SYNC

RSU Clocks (Created from CLK16 Rising Edge)



ADC Clocks (created from CLK16 Falling Edge)



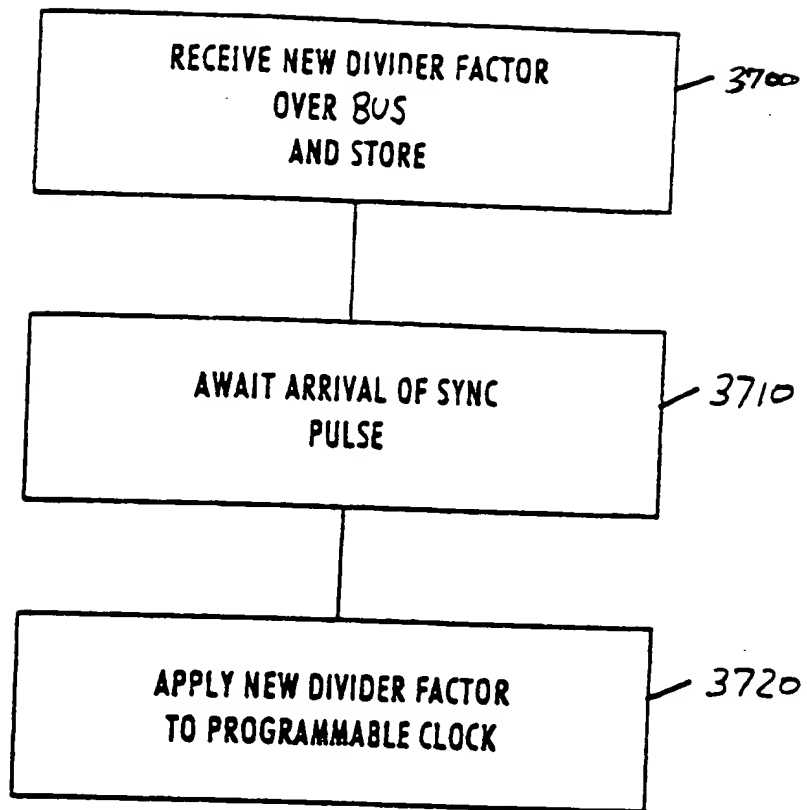


Figure 37